

Preliminary

PNFS/Chimera Metadata Scanning Document

Version 1.0

George Szmuksta

Abstract: This is the documented procedure for performing pnfs/chimera forward and reverse scanning. Scanning is an audit of pnfs/chimera metadata to the enstore file and volume database. Included are the instructions for building a node with hardware specifications and the software needed to perform this task.

Contents

1.) Overview	Overview of the document
2.) Load Operating System.....	Procedural Steps
3.) Making Raid Symbolic Links.....	Procedural Steps
4.) Making Required Enstore Directories.....	Procedural Steps
5.) Install Required Software.....	Procedural Steps
6.) Modify Postgres and pnfs files.....	Procedural Steps
7.) Copy Special Enstore Config.....	Procedural Steps
8.) Restoring Chimera or PNFS DB.....	Procedural Steps
9.) Restoring Enstore DB.....	Procedural Steps
10.) As user enstore setup and start enstore.....	Procedural Steps
11.) Forward Scan.....	Procedural Steps
12.) Reverse Scan.....	Procedural Steps
13.) Output Files.....	Description of output files
14.) Scanning Errors And Warnings.....	Handling errors

Appendix:

15.) Hardware.....	Example stkenscan1 hardware configuration
16.) System Environment.....	Linux and enstore environment
17.) System Processes.....	System processes running on a configured system
18.) Enstore Configuration File.....	Customized enstore conf used on a scanning node

1.) Overview:

Pnfs/chimera scanning is the process of checking the consistency (basically an audit) of tape metadata information (pnfs/chimera) with the enstore volume and file database data. Pnfs and chimera are two different methods of accessing the enstore tape system metadata. Chimera is the new and current method for most enstore user dbs (currently cms is still using pnfs). There is a partial scan and a full scan. The command used in scanning is **enstore scan** (use –help or see online documentation for scanning at http://home.fnal.gov/~zalokar/Enstore_Administrator_Guide10.html).

These scans are run against a locally restored pnfs/chimera and enstore databases (on a local

raid unit). The backups should be from the same day otherwise there could be errors generated because the backups are not from the same time. These scans should **NEVER** be run against the live pnfs/chimera and enstore databases.

The input into a partial scan is a list of files. We will probably will only be doing full scans. There are two methods of full scans. For the forward scan, input is a list of pnfs/chimera mount points. Input into a reverse scan is a list of storage groups.

The scans need to be run on a node that is set up for this purpose. In the **Appendix** there is a description of this node (stkenscan1 is used as an example). Also in the **Appendix** there is a list of hardware, software and the environment required to run these scans. Occasionally when the operating system gets updated or re-purposing an existing node, the internal drives get erased. When this happens the enstore environment needs to be re-installed and configured. The **Software Installed**, **System Environment**, **System Processes** and **Enstore Configuration** sections show how to build a scan node. When the operating system is updated the raid unit can be unplugged so that it does not get erased. Once the node is built the pnfs/chimera and the enstore databases can be restored multiple times without rebuilding the node. It is important to note that when the pnfs/chimera and enstore databases are to be restored, pnfs/chimera should be restored first and enstore if running needs to be stopped. This will happen if the node being used already has pnfs/chimera and enstore installed and is started by **init.d**. The reason for this is that postgresql is started as one of the last steps of the pnfs/chimera restore program. The program will wait if postgresql is already running.

The procedures below are for building a scan node from scratch. If the node has been built for scanning and the aim is restore pnfs/chimera and enstore databases from another day all that needs to be done are steps 8 and 9. Restoring pnfs/chimera database and restoring enstore database.

2.) Procedure For Loading Operating System on Node Using Cobbler if necessary:

Please note that the target node (host being used to run the scan on) has to have a gigabit ethernet connection. Cobbler will not work if the only network connection is 10 Gige. The 10 Gige connection must be unplugged and a gigabit ethernet connection used.

- Login to ssasrv6 as root and open console to target node.
- From the console, or login to target node and shutdown and halt.
- Disconnect raid unit from target node.
- Disconnect second network cable if target node has two connections
- In a web browser access <http://ssasrv2.fnal.gov/cobbler/web>
- User and password are required.
- Select Systems on the left hand side of the page
- Select List
- Click the TARGET NODE in the list
- Check the Netboot enabled box then Save
- Power on The Target node

- Watch the console connection to the target node
- At the login prompt log in as root
- Run the configuration script `./configure_system.sh stk` (or d0, cdf gcc, ssa)
- It will ask for your kerberos user and password
- If `/etc/init.d/3dm2` hangs, login in another window as root and restart it. `/sbin/service 3dm2 restart`.
- When finished plug in the raid unit and second network connection if any.
- If you are unable to ssh in and the node has a second network connection the network bonding config may have to be checked. Temporary edit `/etc/sysconfig/network-scripts/ifcfg-bond0`. Add line after DEVICE, `IPADDR=131.225.13.14` This example is for STK.
- Reboot

3.) Procedure for making required raid symbolic links if they do not exist

Two large raid unit file systems that are attached to the target host are needed to hold the enstore and pnfs/chimera databases. Each file system has two symbolic names defined that are used by the scripts that restore enstore and pnfs dbs and are used by the pnfs scanning programs. They need to be made if they do not exist. The file system for pnfs should have links **srv1** and **diska**. The file system for the enstore databases should have links **srv0** and **diskb**.

If there is only one mount point defined for the entire raid unit, two subdirectories should be made off the mountpoint and the symbolic links made to these subdirectories.

Examples:

Two or more raid unit mountpoints:

pnfs/chimera file system is `/data/data2`:

in / (root) as user root do

- `ln -s /data/data2 /diska`
- `ln -s /data/data2 /srv1`

enstore file system is `/data/data1`

in / (root) as user root do

- `ln -s /data/data1 /diskb`

- ln -s data/data1 /srv0
- Make directory /pnfs and /pnfs/fs

One raid unit mountpoint:

If the operating system was reinstalled you must remove any old postgresql data left over on the raid unit. Remove /srv1/pnfs. and /var/lib/pgsql

4.) Procedure for making required enstore directories if they do not exist

As user root:

- mkdir /srv0/enstore
- chown enstore.enstore /srv0/enstore
- mkdir /srv0/enstore/databases
- chown enstore.enstore /srv0/enstore/databases
- mkdir /srv0/SCANNED
- chown enstore.enstore /srv0/SCANNED
- mkdir /srv0/enstore/databases/enstore-db
- chown products.products /srv0/enstore/databases/enstore-db
- mkdir /srv0/enstore/databases/enstore-journal
- chown enstore.enstore /srv0/enstore/databases/enstore-journal
- As user enstore touch /srv0/enstore/databases/enstore-journal/file.jou
- As user enstore touch /srv0/enstore/databases/enstore-journal/volume.jou

5.) Install required software if necessary

If postgresql-server, pnfs-postgresql, enstore and chimera (dcache and java also required for chimera) are not installed on the target node They need to be installed. Currently if using chimera instead of pnfs, pnfs still has to be installed as the pnfs restore procedure will start pnfs when the restore is complete. It is necessary to stop pnfs and start chimera.

To check if they are installed:

For enstore:

See if required software is installed:

```
root@stkenscan1 enstore]# rpm -qa | grep enstore  
enstore-2.2.2-2.x86_64  
enstore_html-2.0-0.noarch
```

See if pnfs is installed:

```
[root@stkenscan1 enstore]# rpm -qa | grep pnfs  
pnfs-postgresql-3.1.18-1.x86_64
```

See if postgresql is installed:

```
[root@stkenscan1 enstore]# rpm -qa | grep postgres  
postgresql-8.4.3-1PGDG.rhel5.x86_64  
postgresql-libs-8.4.3-1PGDG.rhel5.x86_64  
postgresql-server-8.4.3-1PGDG.rhel5.x86_64  
pnfs-postgresql-3.1.18-1.x86_64  
postgresql-devel-8.4.3-1PGDG.rhel5.x86_64
```

To install **postgresql**, as user root:

- yum –enablerepo=enstore install postgresql-server
- cd to /var/lib
- remove postgresql data directory – **rm -rf pgsql**

To install **pnfs**, as user root:

- yum –enablerepo=enstore install pnfs-postgresql

To install **enstore**, as user root:

- wget ftp://ssasrv1.fnal.gov/en/sl5x/noarch/install_enstore_rpm.sh
- chmod +x install_enstore_rpm.sh
- Get a copy of offline.conf from another node or from the Appendix
- copy offline.conf to /home/enstore
- ./install_enstore_rpm.sh
- **Temporary** – cvs update /opt/enstore/tools and /opt/enstore/sbin
- cd /opt/enstore/tools
- cvs update -r production

- cd /opt/enstore/sbin
- cvs update -r production

When the install asks for configuration file, point to /home/enstore/offline.conf (see Step 7 Copy or make copy of the enstore configuration file). When it asks for email recipient use your email. Take defaults for the rest of the questions.

See if chimera is installed:

- 1.) See if dcache is installed:

```
rpm -qa | grep dcache
```

- 2.) If not installed, install dcache:

To find out what version of dcache to install find out what is the current version being used in production. Login as enstore on a dcache head node (example fndca) and get the dcache verion number on the dcache head node run /opt/d-cache/bin/dcache version. Should look something like this:

```
enstore@fndca3a ~]$ /opt/d-cache/bin/dcache version
production-1.9.5-28
[enstore@fndca3a ~]$
```

Using a web browser go the the dcache site <http://www.dcache.org/downloads> and look for the proper version. Chances are very good that the needed version is in the “unsupported” (too old) folder. After the version of the dcache server has been located run the rpm to install it:

```
rpm -i http://www.dcache.org/downloads/1.9/unsupported/dcache-server-1.9.5-28.noarch.rpm
```

To check after the install run rpm -qa | grep dcache

```
root@stkendm5a pgsql]# rpm -qa | grep dcache
dcache-server-1.9.5-28.noarch
```

- 3.) Check to see if java is installed:

```
yum list all | grep java
```

- 4.) If java is not installed, install java:

yum install java

Copy the default dcache config file.

```
cd /opt/d-cache/config  
cp /opt/d-cache/etc/dCacheSetup.template dCacheSetup
```

Change user from postgres to enstore in chimera_config.xml

```
sed -i 's/"postgres"/"enstore"/g' /opt/d-cache/config/chimera-config.xml
```

Copy chimera-nfs-run.sh into init.d

```
cp /opt/d-cache/libexec/chimera/chimera-nfs-run.sh /etc/init.d/
```

6) Add chimera-nfs-run.sh to services and enable.

```
chkconfig --add chimera-nfs-run.sh  
chkconfig chimera-nfs-run.sh on
```

7) If pnfs is on in chkconfig turn it off.

```
chkconfig pnfs off
```

8) cat > /etc/exports << EOF
/ localhost(rw)
/pnfs localhost(rw)
/fs localhost(rw)
EOF

6.) Make modifications and additions to postgesql and pnfs

Copy missing required file.

Copy file pnfs_wrapper to /etc/init.d . This file starts pnfs after the system is booted.

As root cd /etc/init.d
cp /opt/enstore/tools/pnfs_wrapper .

Copy postgresql_check into /opt/pnfs/tools

As root cd cd /opt/pnfs/tools
cp /opt/enstore/tools/postgres_check .

Modify /etc/init.d/postgresql. This is the file that starts postgesql after the system has been booted.

As root cd /etc/init.d
cp postgresql postgres~ or some other name.

Edit postgresql file and do the following:

Find line chown postgres:postgres "\$PGLOG"
Change owner chown enstore.enstore "\$PGLOG"

Find line chown postgres:postgres "\$PGDATA"
Change owner chown enstore.enstore "\$PGDATA"

Find line chown postgres:postgres "\$PGLOG"
Change owner chown enstore.enstore "\$PGLOG"

Find line chown postgres:postgres "\$PGDATA/pg_log"
Change owner chown enstore.enstore "\$PGDATA/pg_log"

Find line \$SU -l postgres -c "\$PGENGINE/postmaster -p '\$PGPORT' -D '\$PGDATA' \${PGOPTS} &">> "\$PGLOG" 2>&1 </dev/null

Change login \$SU -l enstore -c "\$PGENGINE/postmaster -p '\$PGPORT' -D '\$PGDATA' \${PGOPTS} &">> "\$PGLOG" 2>&1 </dev/null

Find line \$SU -l postgres -c "\$PGENGINE/pg_ctl stop -D '\$PGDATA' -s -m fast" > /dev/null 2>&1 </dev/null

Change login \$SU -l enstore -c "\$PGENGINE/pg_ctl stop -D '\$PGDATA' -s -m fast" > /dev/null 2>&1 </dev/null

Find line \$SU -l postgres -c "\$PGENGINE/initdb --pgdata='\\$PGDATA' --auth='ident'" >> "\$PGLOG" 2>&1 </dev/null

Change login \$SU -l enstore -c "\$PGENGINE/initdb --pgdata='\\$PGDATA' --auth='ident'" >> "\$PGLOG" 2>&1 </dev/null

Find line \$SU -l postgres -c "\$PGENGINE/pg_ctl reload -D '\$PGDATA' -s" > /dev/null 2>&1 </dev/null

Change login \$SU -l enstore -c "\$PGENGINE/pg_ctl reload -D '\$PGDATA' -s" > /dev/null 2>&1 </dev/null

Save the file.

Initialize Postgresql DB:

As root :

```
root@stkendm5a ~]# /sbin/chkconfig --list postgresql
postgresql      0:off    1:off    2:off    3:off    4:off    5:off    6:off
```

```
root@stkendm5a /sbin/chkconfig postgresql on
```

Init Postgresql:

After a reload of the operating system if the system was not reloaded you do not have to do this :

```
/sbin/service postgresql initdb
```

Add pnfs_wrapper to chkconfig and turn on

```
/sbin/chkconfig -add pnfs_wrapper
```

```
/sbin/chkconfig pnfs_wrapper on
```

7.) Copy or make a copy of the enstore configuration file offline.conf

A Special cut down version of an enstore configuration file called offline.conf. This file contains configuartion and info server information and stub files for all of the library managers from all of the live enstore systems. This allows any enstore and matching pnfs dbs to be used without changing the configuration.

This file can be copied from a scan node or a file can be made from the listing in this document. It needs to be copied to /opt/enstore/etc/, Changes need to be made to this file before it can be used with enstore.

The changes are:

DB_host and info_server_host need to be set to the full qualified name of the node that you will be sing as the scan node. Example if you are using stkenscan1 then set DB_host and info_server_host to stkenscan1.fnal.gov.

If defaults were take during the enstore software installation environmental variable ENSTORE_CONFIG_FILE needs to changed to point to offline.config. This change and other changes to the ENSTORE environmental variables are made to the file /home/enstore/site_specific/config/setup-enstore file.

As user enstore edit /home/enstore/site_specific/config/setup-enstore

The changes are:

ENSTORE_MAIL=georges@fnal.gov

ENSTORE_HOME=/home/enstore

ENSTORE_CONFIG_HOST=stkenscan1.fnal.gov
ENSTORE_CONFIG_PORT=7500
ENSTORE_CONFIG_FILE=/opt/enstore/etc/offline.conf
ENSTORE_DIR=/opt/enstore

Make a symbolic link in /home/enstore called enstore which should point to /opt/enstore:

ln -s /opt/enstore enstore

8.) Procedure for restoring pnfs/chimera on a scan node

- Make sure that the same postgresql version is installed on stkenscan1 (it is currently the same as on d0en, cdfen, stken, cms uses postgresql 8.3)
- Logging in as enstore to stkenscan1.
- If enstore is running stop it. --- as user *enstore* **enstore stop**.
- If the postmaster process for the enstore database does not end it must be killed.
- **export ENSTORE_CONFIG_HOST=d0ensrv2n.fnal.gov** (or stkensrv2, cdfensrv2n).
- Then become a root (This requires a kerberos ticket): run **ksu**
- run this command to restore pnfs database on stkenscan1. Example d0:
/opt/enstore/tools/pnfs_db_restore.py -s d0en
- To get help for the command run without target system
/opt/enstore/tools/pnfs_db_restore.py

Usage: /opt/enstore/tools/pnfs_db_restore.py -s [--system=] -t [backup_time=]
allowed systems: ['d0en', 'sdss', 'cms', 'eag', 'stken', 'd0en', 'cdfen']
specify timestamp YYYY-MM-DD to get backup up to certain date

It takes some time to run this program (45 minutes). After it completes you should have pnfs started and mounted.

Output looks like this:

```

Creating recovery.conf: restore_command = '/opt/enstore/sbin/gettkt && /opt/enstore/sbin/enrcp
131.225.215.14:/srv3/enstore/backups/pnfs-backup.xlogs/%f.Z %p.Z && uncompress %p.Z'
restore_command = '/opt/enstore/sbin/gettkt && /opt/enstore/sbin/enrcp
131.225.215.14:/srv3/enstore/backups/pnfs-backup.xlogs/%f.Z %p.Z && uncompress %p.Z'
Starting postgresql service: [ OK ]
Starting DB server
Waiting for DB server
DB server is ready
Starting pnfs: /sbin/service pnfs_wrapper start
pnfs_wrapper: checking postgresql: [ OK ]
Starting pnfs services (PostgreSQL version):
Shmcom : Installed 10 Clients and 50 Servers
Starting database server for admin (/diska/pnfs/db/admin) ... O.K.
Starting database server for cdfen (/diska/pnfs/db/cdfen) ... O.K.
Starting database server for GI (/diska/pnfs/db/GI) ... O.K.
Starting database server for CA (/diska/pnfs/db/CA) ... O.K.
Starting database server for PR (/diska/pnfs/db/PR) ... O.K.
Starting database server for GJ (/diska/pnfs/db/GJ) ... O.K.
Starting database server for PS (/diska/pnfs/db/PS) ... O.K.
Starting database server for CB (/diska/pnfs/db/CB) ... O.K.
Starting database server for ssa_test (/diska/pnfs/db/ssa_test) ... O.K.
Starting database server for PT (/diska/pnfs/db/PT) ... O.K.
Starting database server for SM (/diska/pnfs/db/SM) ... O.K.
Starting database server for PU (/diska/pnfs/db/PU) ... O.K.
Starting database server for PV (/diska/pnfs/db/PV) ... O.K.
Starting database server for PW (/diska/pnfs/db/PW) ... O.K.
Starting database server for PX (/diska/pnfs/db/PX) ... O.K.
Starting database server for PY (/diska/pnfs/db/PY) ... O.K.
Starting database server for PZ (/diska/pnfs/db/PZ) ... O.K.
Starting database server for GK (/diska/pnfs/db/GK) ... O.K.
Starting database server for XX (/diska/pnfs/db/XX) ... O.K.
Starting database server for XY (/diska/pnfs/db/XY) ... O.K.
Starting database server for CC (/diska/pnfs/db/CC) ... O.K.
Starting database server for CD (/diska/pnfs/db/CD) ... O.K.
Starting database server for SN (/diska/pnfs/db/SN) ... O.K.

```

Starting database server for NB (/diska/pnfs/db/NB) ... O.K.
Starting database server for NS (/diska/pnfs/db/NS) ... O.K.
Starting database server for NT (/diska/pnfs/db/NT) ... O.K.
Starting database server for NULL (/diska/pnfs/db/NULL) ... O.K.
Starting database server for SO (/diska/pnfs/db/SO) ... O.K.
Starting database server for MC (/diska/pnfs/db/MC) ... O.K.
Starting database server for MD (/diska/pnfs/db/MD) ... O.K.
Starting database server for NN (/diska/pnfs/db/NN) ... O.K.
Starting database server for OO (/diska/pnfs/db/OO) ... O.K.
Starting database server for TT (/diska/pnfs/db/TT) ... O.K.
Starting database server for sl8500 (/diska/pnfs/db/sl8500) ... O.K.
Waiting for dbservers to register ... Ready
Starting Mountd : pmountd
Starting nfsd : pnfsd

DONE

If the ***postgresql server does not start*** check the postgresql startup log in /var/lib/pgsql/pgstartup.log for error messages. After it starts the ***other log file*** is in /srv1/postgresql-log.

Please note that pnfs restore has a date option to get a backup from a certain date. This option does not exist in the enstore DB restore.

If you perform multiple pnfs restores be aware that old restore data does not get erased. It gets renamed. Check the space on /srv1 and erase unneeded pnfs restore data.

Stop pnfs - -

/sbin/service pnfs stop

Umount /pnfs/fs - -

umount -l /pnfs/fs

Start Chimera - -

/sbin/service chimera-nfs-run.sh start

Mount /pnfs/fs - -

mount localhost:/fs /pnfs/fs

9.) Procedure for restoring Enstore Databases on a scan node

* Login as enstore. point ENSTORE_CONFIG_HOST to the config host of the system you are restoring. . For example d0en

export ENSTORE_CONFIG_HOST=d0ensrv2n.fnal.gov

This will point to the system where the backup file will come from.

- If enstore is running stop it. --- as user *enstore* ***enstore stop***

* Become root (this requires a kerberos ticket):

Run: ***ksu***

* Create an empty database – run:

python ~enstore/enstore/sbin/create_database.py enstroredb

If a database already exists it is not necessary to create a new database. You just need to restore the database.

To get help run command without a specified database:

python ~enstore/enstore/sbin/create_database.py

* Restore the selected enstore database into the empty database on stkenScan1:

python ~enstore/enstore/sbin/dump_restore_database.py -r -b enstroredb

and wait until it completes. This takes up to 1.5 hours to complete.

Output Looks like this:

```
[root@stkendm5a enstore]# python ~enstore/enstore/sbin/dump_restore_database.py -r -b enstroredb
copying: /usr/bin/rsync -e rsh cdfensrv3n.fnal.gov:/srv3/enstore/backups/enstore-
backup/dbase.1322061002.5882699/enstroredb.dmp .
2011-11-23 09:31:41 : Got backup file enstroredb.dmp
2011-11-23 09:31:41 : About to drop database enstroredb running on localhost port 8888
2011-11-23 09:31:41 : Sleep 10 seconds
2011-11-23 09:31:51 : Dropping
2011-11-23 09:32:50 : Successfully dropped database enstroredb
2011-11-23 09:32:51 : Successfully created database enstroredb
```

```
2011-11-23 09:32:51 : Restoring database enstорedb  
2011-11-23 10:16:35 : Done  
[root@stkendm5a enstore]#
```

The Pnfs restore command in the Restore PNFS DB section has a date option to use a backup from a certain date to restore. The enstore DB restore does not have this option.

10.) As user enstore setup and start enstore

To setup enstore, source setup-enstore file:

```
./home/enstore/site_specific/config/setup-enstore
```

Start the configuration server:

```
[enstore@stkenscan1 ~]$ enstore start --just configuration_server  
Checking configuration_server.  
Starting configuration_server
```

Start the info server:

```
[enstore@stkenscan1 ~]$ enstore start --just info_server  
Got error while trying to obtain configuration: ('KEYERROR', "Configuration Server: no such name: 'log_server'")  
Got error while trying to obtain configuration: ('KEYERROR', "Configuration Server: no such name: 'alarm_server'")  
Checking info_server.  
Starting info_server: 131.225.13.10:7777  
[enstore@stkenscan1 ~]$
```

The error messages are normal as we do not have a log or alarm server defined.

11.) Forward Scan

The scanning documentation from the developers is online”

http://home.fnal.gov/~zalokar/Enstore_Administrator_Guide10.html

All scans are started in the /srv0/SCANNED directory as user enstore. This procedure will refer to full scans. Scans usually follow pnfs and enstore database restores though you can scan multiple times on the same pnfs and enstore databases that were previously used. Forward scans use a list of

pnfs mountpoints as input. The list can be created by using the *mount_point* script located in /home/enstore/enstore/sbin (/opt/enstore/sbin). The output goes to std out so you must redirect the output to a file. Example: *mount_points* > *mount_point_file.txt*. To start a forward scan:

```
[enstore@stkendm5a SCANNED]$ mount_points
/pnfs/fs/usr/cdfen
/pnfs/fs/usr/Migration
/pnfs/fs/usr/ssa_test
```

```
[enstore@stkendm5a SCANNED]$ mount_points > cdfen_mountpoints_112311
```

```
[enstore@stkendm5a SCANNED]$ forward_scan cdfen_mountpoints_112311 | tee
current_scan_output
```

Output looks like this:

```
$ENSTORE_CONFIG_HOST = stkendm5a.fnal.gov
Starting: Wed Nov 23 10:23:21 CST 2011
Starting to scan /pnfs/fs/usr/cdfen at Wed Nov 23 10:23:21 CST 2011
The 'cdf' storage group has 8226941 files to check.
time enstore scan /pnfs/fs/usr/cdfen
```

12.) Reverse Scan

A reverse scan is also started in the /srv0/SCANNED directory. It uses a storage group list as input. This list can be made with the *storage_groups* script in /home/enstore/enstore/sbin or /opt/enstore/sbin. It works the same way as the *mount_points* script.

Example:

```
storage_groups > storage_groups.txt.
```

To start a reverse scan after the *storage_groups* input file is made:

As user **enstore**:

- Make sure enstore has been setup using the offline.config file.
- Check this by doing an *env | grep ENSTORE*
- If not source the enstore setup file *./home/enstore/site_specific/config/setup-enstore* See the example in the Forward Scan section above.
- *cd /srv0/SCANNED*
- *reverse_scan storage_group_file.txt | tee current_scan_output*

13.) Output file

All output files reside in directory /srv0/SCANNED/SCAN_MM_DD_YYYY. Where the MM_DD_YYY is the month day and year when the scan was started.

The format of the file names is *storage-group_scan-type[.info-type]*

Examples:

cdf_reverse
cdf_forward.warning
cdf_reverse.error

Info types:

astro_reverse.error	- errors from the reverse scan of storage group astro
astro_reverse.warning	- warnings from the reverse scan of storage group astro.
auger_reverse	- all output messages from a reverse scan of storage group auger.
reverse_scan.status	- status messages for the current reverse scan.
forward_scan.status	- status messages for the current forward scan.

14.) Scanning Errors And Warnings

ERRORS:

“found temporary file”

EXAMPLE: /pnfs/fs/usr/dzero/db5/derivedDetector/csg-p20.18.02b/dzero/thumbnail/fix2/0004/.nfs0000000002e765980000237b ... found temporary file ...
ERROR

- These should be removed using rm.
- If “invalid directory entry” also appears, it means that entry in the directory listing does not point to an existing file but to a ghost file.

“missing file”

EXAMPLE: /pnfs/fs/usr/dzero/db5/derivedSimulated/csg-p21.21.00-v1/dzero/root-tree/generic/0030/CAF-MCv1-150443-MERGE_p20.17.02_NumEv-10000_hl+z-incl_sm.n_higgs_mcp20_ccin2p3_150443_1341581681832286411248093703_p21.21.00_dq_dq.root .. missing file ... ERROR

- The file has no size, no layer 1, 2 or 4. This can be checked by doing an ***ls -l*** of the file on the storage group's pnfs server (example: stkensrv1n). It should give no size.
- To check the layers of the file use the “***pnfs –layer file_path_and_name layer_number***”.
- To see if the file was written to tape try to get the bfid by using ***enstore file –bfid file_name*** if no bfid is returned the file was not written to tape.
- The experiment should be notified.
- The file should be removed (use ***rm*** as ***root*** on ****srv1n carefully***).
- If there are many of these errors a list can be made from the .error file. Use 'cat ***error_file_name.error | grep “missing file” | cut -d” “ -f1 > missing_file.txt***'.
- Edit the file with ***vi*** and get into ex mode – press the ***ESC*** key and type with no quotes “***:1,\$ s/pnfs/rm -f \pnfs/g***” then save. This will insert a “***rm -f***” before each file name and this can be used as a script to delete the files on ***srv1***. If you want it to prompt you for a yes or no before each delete leave off the “***-f***”. Then turn execute permission on – ***chmod +x missing_file.txt***.

“not a storage filesystem file or directory”

EXAMPLE: /diskb/SCANNED/9940BCLN ... not a storage filesystem file or directory ... ERROR

The file or directory is not valid for scan input and it will not be scanned. Ignore.

“not in db”

EXAMPLE: /pnfs/fs/usr/des/Archive/raw/imSim7_SN-2013-02-05/src/decam-bias-3.fits.fz ... not in db ... ERROR

“storage filesystem entry exists ... deleted(yes)”

EXAMPLE: /pnfs/fs/usr/des/Archive/raw/imS/pnfs/fs/usr/eagle/.m.srv3-enstore-backups-enstore-backup-dbase.1292591402.4831841-volume.tar.gz ... storage filesystem entry exists ... deleted(yes) ... ERROR

“missing layer 1 ... missing layer 4”

EXAMPLE: pnfs/fs/usr/eagle/dcache-tests/10.dcache_page_gkftp.12451.1.1_stream ... missing layer 1 ... missing layer 4 ... ERROR

“sfsid() renamed”

EXAMPLE: /pnfs/fs/usr/eagle/d0en-backups/2011/12/06/07/.m.srv3-enstore-backups-enstore-backup-ACC-DST-accounting.backup.01 ... sfsid(00010000000000003BF9AD8, 00010000000000003BF9AD8, 00010000000000003B9A6A0) ... renamed(/pnfs/fs/usr/eagle/d0en-backups/2011/12/06/07/.m.srv3-enstore-backups-enstore-backup-ACC-DST-accounting.backup.01, /pnfs/eagle/d0en-backups/2011/12/06/07/srv3-enstore-backups-enstore-backup-ACC-DST-accounting.backup.01) ... ERROR

“missing layer 4 crc”

EXAMPLE: /pnfs/fs/usr/ktev/isd_test/NULL/NQDM05.dat ... missing layer 4 crc ... no such library (null1) ... ERROR

“invalid directory entry”

EXAMPLE:

/pnfs/fs/usr/Migration/miniboone/OpenRootTree/Strobe/run0014000_0014999/run0014070_0014079/b oone_0014079_0017.root.strobe ... invalid directory entry ... ERROR

These errors occur when a hard link succeeds to create the new directoryentry, but fails to update the link count. This is a similar situation to the "duplicate entry" errors described below.

To confirm the problem:

```
# cd <broken dir>
# ls | grep <basename> #will take a while for large directories.
stat <basename>
```

If the "ls | grep" says the file exists, but stat says "No such file or directory" then this is a corrupted files (A.K.A. ghost file).

```
# ls | grep 0_p18.05.00.root
0_p18.05.00.root
# stat 0_p18.05.00.root
stat: cannot stat `0_p18.05.00.root': No such file or directory
```

To fix the problem (as root and the broken directory is the current working directory):

```
# cd <broken dir>  
  
# /opt/pnfs/tools/sclient rmdirentry 1122 `enstore pnfs --id .` <basename>
```

For example:

```
# /opt/pnfs/tools/sclient rmdirentry 1122 `enstore pnfs --id .` 0_p18.05.00.root
```

“no layer 2 size”

EXAMPLE: /pnfs/fs/usr/test/NeST/NULL/srmtest/repeat/oct06_4667.small ... no layer 2 size ... missing layer 4 crc ... no layer 2 crc ... no such library (null1) ... ERROR

“missing layer 4 crc ... no layer 2 crc”

EXAMPLE: /pnfs/fs/usr/test/NeST/NULL/srmtest/repeat/oct06_4667.small ... no layer 2 size ... missing layer 4 crc ... no layer 2 crc ... no such library (null1) ... ERROR

“duplicate entry”

EXAMPLE:

/pnfs/fs/usr/Migration/miniboone/AuxData/LMC/run0014000_0014999/run0014340_0014349 ...
duplicate entry(boone 0014340 0002.lmc) ... ERROR

If you list the directory with the duplicate error you will see something like:

1s

all all all

To see what PNFS see internally use the scandir.sh script.

```
# /opt/pnfs/tools/scandir.sh  
00090000000000000000163470 00090000000000000000163430 0000000000000000 0 all  
00090000000000000000163470 000900000000000000003525B8 0000000000000200 1 all  
00090000000000000000163470 000900000000000000003525B8 0000000000000200 2 all
```

- The first column lists the PNFS IDs of the directory bucket that each file entry is located in.
 - The second column lists the PNFS ID for the file entry in the directory.
 - The third column is PNFS specific information.

- The fourth column is the position on the file entry within the directory bucket.
- The fifth column is the filename of the file in this directory.

There is a serious complication when different files might be hidden behind the first entry in the directory. The first one listed is the path that gets used by UNIX and Enstore commands. To illustrate this obtain the PNFS ID from the filename. The following example shows how it matches that of the first line of scandir.sh output.

```
# enstore pnfs --id all
```

```
00090000000000000000163430
```

Also, the link count might be different from the number of hard links there really are. In the following "stat" examples the link counts are consistent, but the hidden "all" entries are files not directories.

```
# stat `.(access)(0009000000000000163430)`
File: `.(access)(0009000000000000163430)'
Size: 512 Blocks: 1 IO Block: 512 Directory
Device: ah/10d Inode: 152450096 Links: 1
Access: (0755/drwxr-xr-x)Uid: ( 7816/ sam) Gid: ( 0/ root)
Access: 2006-03-24 10:36:01.000000000 -0600
Modify: 2006-03-24 10:36:01.000000000 -0600
Change: 2001-06-12 14:51:51.000000000 -0500
```

```
# stat `.(access)(00090000000000003525B8)`
File: `.(access)(00090000000000003525B8)'
Size: 0 Blocks: 0 IO Block: 512 Regular File
Device: ah/10d Inode: 154478008 Links: 2
Access: (0644/-rw-r--r--)Uid: ( 7816/ sam) Gid: ( 1507/ e740)
Access: 2001-09-15 00:50:07.000000000 -0500
Modify: 2001-09-15 00:50:07.000000000 -0500
Change: 2001-09-15 00:50:07.000000000 -0500
```

If necessary (the example does not need this) modify link count (assuming CWD is the broken directory):

```
# /opt/pnfs/tools/sclient modlink 1122 <file id> <link diff>
```

"link diff" will be negative to shrink the link count.

If necessary add or remove the links (assuming CWD is the broken directory):

```
# /opt/pnfs/tools/sclient rmdirentry 1122 `enstore pnfs --id .` <name>
```

```
# /opt/pnfs/tools/sclient adddirentry 1122 `enstore pnfs --id .` <name> <pnfsid>
```

For this example, to create new directory entries for the unaccessible "all" files in the example:

```
# /opt/pnfs/tools/sclient adddirentry 1122 `enstore pnfs --id .` all2 \
000900000000000000003525B8

# /opt/pnfs/tools/sclient adddirentry 1122 `enstore pnfs --id .` all3 \
000900000000000000003525B8
```

At this point the output from scandir.sh is:

```
# /opt/pnfs/tools/scandir.sh

0009000000000000000045ED60 000900000000000000003525B8 0000000000000000 0 all2
0009000000000000000045ED68 000900000000000000003525B8 0000000000000000 0 all3
00090000000000000000163470 00090000000000000000163430 0000000000000000 0 all
00090000000000000000163470 000900000000000000003525B8 0000000000000200 1 all
00090000000000000000163470 000900000000000000003525B8 0000000000000200 2 all
```

Continuing to use the output from scandir.sh; we can not just use rm on "all" here¹. We need to remove the extraneous directory entries pointing to "all." The rmdirentrypos command removes directory entries based on their position within the directory.

```
# /opt/pnfs/tools/sclient rmdirentrypos 1122 00090000000000000000163470 \
000900000000000000003525B8 1

# /opt/pnfs/tools/sclient rmdirentrypos 1122 00090000000000000000163470 \
000900000000000000003525B8 2
```

At this point the output from scandir.sh is:

```
# /opt/pnfs/tools/scandir.sh

0009000000000000000045ED60 000900000000000000003525B8 0000000000000000 0 all2
0009000000000000000045ED68 000900000000000000003525B8 0000000000000000 0 all3
00090000000000000000163470 00090000000000000000163430 0000000000000000 0 all
```

Note: The output from ls may be old due to file system caching (especially on Linux):

```
# ls
all all all2 all3
```

To fix this:

```
# umount /pnfs/fs && mount /pnfs/fs
```

And ls will then give:

```
# ls
all all2 all3
```

“no such library”

EXAMPLE: /pnfs/fs/usr/test/NeST/NULL/srmtest/repeat/oct06_4667.small ... no layer 2 size ... missing layer 4 crc ... no layer 2 crc ... no such library (null1) ... ERROR

“*size*”

EXAMPLE: /pmfs/mist/NULL/1KB_093 ... size(1024, 0, 1024) ... ERROR

The file DB and layer 4 sizes agree, but the stat() size (in the middle) is zero. If the size mismatch errors do not fit this pattern then a more detailed investigation needs to be done.

To fix this from the command line:

```
# enstore pnfs --size <filename> <size>
```

“*deleted*”

EXAMPLE:

This likely means that the offline pnfs server still contains the file, but: the file has been removed from the production pnfs server, delfile.py marked the file deleted and the scan found this discrepancy with the offline pnfs server.

If a rerun of the scan of these files on the production pnfs server gives "does not exist" then the user deleted the file and everything is okay. If the file rescans as OK there also is nothing else to do. If the error remains "deleted" then there is a conflict between the Enstore and PNFS databases. In most cases (but not all) the simplest thing is to

unmark the file as deleted in the file DB.

```
$ enstore file --bfid <BFID> --deleted no
```

“*does not exist*”

The file has been removed from PNFS. There is nothing to do.

“*reused pnfsid (reverse scan)*”

In all likelihood the file received an error while writing to tape and a retry succeeded. The files in question should be marked as deleted.

```
$ enstore file --bfid <BFID> --deleted yes
```

Take care to make sure that these BFIDs are not for multiple copies; multiple copies do share PNFS IDs with the current primary file. An early version of enstore scan did not check if a file was a

duplicate before giving this error.

“parent id”

The most common cause of this error is using mv to move a file to another directory and PNFS only partially updates its internal metadata. An incorrect parent ID will cause the `enstore pnfs -path <pnfsid>` command to break, because it uses parent IDs to determine the current path of the file.

The tricky part of this error is that it may be a valid file. If there are two hard links to the i-node from different directories one of the pathnames will give this error while the other will not.

Once it has been decided if/when that the parent ID is wrong, and not a second hard link use the following to change the parent ID of the specified PNFS ID.

```
# /opt/pnfs/tools/sclient chparent 1122 <objectPnfsID> <newParentID>
```

WARNINGS:

“younger than 1 day”

This is a warning. It says that the file is very new and has not finished writing to tape. Check the file again after a day or so

“missing the original of link”

EXAMPLE:

```
/pnfs/fs/usr/fermigrid/volatile/fermilab/1324884481-storage-probe-test-file-remote.15144 ... no layer 2  
size ... WARNING
```

The target of the link is not found. The owner of the link should be notified. Especially, if the link points to files not in the PNFS namespace

“missing layer 4 drive ... missing layer 4 crc “

EXAMPLE:

/pnfs/fs/usr/archive/d0en/mover/files.list ... missing layer 4 drive ... missing layer 4 crc ... WARNING

“no layer 2 size”

EXAMPLE:

/pnfs/fs/usr/fermigrid/volatile/fermilab/1324884481-storage-probe-test-file-remote.15144 ... no layer 2 size ... WARNING

Appendix:

15.) Hardware

- 1.) Using stkenscan1 as an example.
- 2.) A rack mounted PC with twin 2.83GHz Xeon processors.
- 3.) 8 GB memory.
- 4.) Linux version 2.6.18-238.5.1.el5.
- 5.) A Nexan Satablade raid unit is fiber attached.
- 6.) Raid is mounted as 2 volumes – 921Gb Raid 0 /diska and 461Gb Raid 5 /diskb
- 7.) Symbolic links srv0 → /diska and /srv1 → /diskb.

16.) System environment

The following directories need to be made if they do not exist and ownerships set. Make sure the symbolic links for /srv0 and /srv1 are made as the restored enstore database files will be in /srv0. The enstore configuration file will point to these database files. The restored pnfs files will be in /srv1.

mkdir /srv0/enstore

Environmental variables should look like this after everything is installed:

```

ENSTORE_OUT=/var/log/enstore
HOSTNAME=stokenscan1.fnal.gov
ENSTORE_MAIL=georges@fnal.gov
TERM=xterm
SHELL=/bin/bash
HISTSIZE=10000
SSH_CLIENT=131.225.81.58 46699 22
GADFLY_GRAMMAR=/opt/enstore/gadfly
PYTHONUNBUFFERED=x
PYTHON_DIR=/opt/enstore/Python
ENSTORE_HOME=/home/enstore
SSH_TTY=/dev/pts/0
USER=enstore
LS_COLORS=no=00:fi=00:di=00:34:ln=00:36:pi=40:33:so=00:35:bd=40:33:01:cd=40:33:01:or=01:05:37:41:mi=01:05:37:41:ex=00:32:*.cmd=00:32:*.e
xe=00:32:*.com=00:32:*.btm=00:32:*.bat=00:32:*.sh=00:32:*.csh=00:32:*.tar=00:31:*.tgz=00:31:*.arj=00:31:*.taz=00:31:*.lzh=00:31:*.zip=00:31:*.z=
00:31:*.Z=00:31:*.gz=00:31:*.bz2=00:31:*.bz=00:31:*.tz=00:31:*.rpm=00:31:*.cpio=00:31:*.jpg=00:35:*.gif=00:35:*.bmp=00:35:*.xbm=00:35:*.xpm=
00:35:*.png=00:35:*.tif=00:35:
ENSTORE_CONFIG_HOST=stokenscan1.fnal.gov
MAIL=/var/spool/mail/enstore
PATH=/usr krb5/bin:/opt/enstore/Python/bin:/opt/enstore/sbin:/opt/enstore/bin:/opt/enstore/tools:/opt/enstore/SWIG:/fnal/ups/prd/ups/v4_7_2/Linux-
2/bin:/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin
UPS_SHELL=sh
INPUTRC=/etc/inputrc
PWD=/home/enstore/scan_stuff
LANG=en_US.UTF-8
PYTHONINC=/opt/enstore/Python/include/python2.6
SETUP_UPS=ups v4_7_2 -f Linux+2 -z /local/ups/db
UPS_DIR=/fnal/ups/prd/ups/v4_7_2/Linux-2
KRB5CCNAME=FILE:/tmp/krb5cc\_5744\_CzQJOu5312
ENSTORE_CONFIG_PORT=7500
SHLVL=1
HOME=/home/enstore
ENSTORE_CONFIG_FILE=/opt/enstore/etc/offline.conf
SWIG_LIB=/opt/enstore/SWIG/swig_lib
FTT_DIR=/opt/enstore/FTT
PYTHONPATH=/opt/enstore:/opt/enstore/src:/opt/enstore/modules:/opt/enstore/HTMLgen:/opt/enstore/PyGreSQL
PYTHONLIB=/opt/enstore/Python/lib/python2.6
LOGNAME=enstore
CVS_RSH=ssh
SSH_CONNECTION=131.225.81.58 46699 131.225.13.10 22
FARMLETS_DIR=/usr/local/etc/farmlets
LESSOPEN=|/usr/bin/lesspipe.sh %s
ENSTORE_DIR=/opt/enstore
SWIG_DIR=/opt/enstore/SWIG
SETUPS_DIR=/fnal/ups/etc
DISPLAY=localhost:10.0
PRODUCTS=/local/ups/db:/fnal/ups/db
MACH_OS=Linux
G_BROKEN_FILERAMES=1
HISTTIMEFORMAT=%b %e %T
KRBTICKFILE=/tmp/tkt5744_CqcrvN
_=bin/env
OLDPWD=/home/enstore

```

17.) System processes:

These are the system processes that should be running when everything is installed and configured properly and after the enstore and pnfs restores are completed. This example has a D0 pnfs and enstore db.

```

enstore 24577 0.0 1.7 292532 24577 1 pts/0 S Aug16 00:00:02 python /opt/enstore/sbin/info_server
enstore 2950 0.0 0.1 388876 2950 1 ? S Aug09 00:00:52 /usr/bin/postmaster -p 5432 -D /diska/pnfs/db/pgsql-data
enstore 2954 0.0 0.0 389140 2954 1 ? Ss Aug09 00:00:00 postgres: writer process
enstore 2955 0.0 0.0 389008 2955 1 ? Ss Aug09 00:00:00 postgres: wal writer process
enstore 3044 0.0 0.0 390260 3044 1 ? Ss Aug09 00:00:00 postgres: enstore admin [local] idle
enstore 3048 0.0 0.0 390260 3048 1 ? Ss Aug09 00:00:00 postgres: enstore NULL [local] idle
enstore 3052 0.0 0.0 390260 3052 1 ? Ss Aug09 00:00:00 postgres: enstore db5 [local] idle
enstore 3056 0.0 0.0 390220 3056 1 ? Ss Aug09 00:00:00 postgres: enstore ssa_test [local] idle
enstore 3060 0.0 0.0 390220 3060 1 ? Ss Aug09 00:00:00 postgres: enstore archive [local] idle
enstore 3064 0.0 0.0 390212 3064 1 ? Ss Aug09 00:00:00 postgres: enstore run1 [local] idle
enstore 3068 0.0 0.0 390212 3068 1 ? Ss Aug09 00:00:00 postgres: enstore runi-a [local] idle
enstore 3072 0.0 0.0 390212 3072 1 ? Ss Aug09 00:00:00 postgres: enstore sam-mammoth [local] idle
enstore 3076 0.0 0.0 390212 3076 1 ? Ss Aug09 00:00:00 postgres: enstore runi-b [local] idle
enstore 3080 0.0 0.0 390212 3080 1 ? Ss Aug09 00:00:00 postgres: enstore sam-m2 [local] idle
enstore 3084 0.0 0.0 390212 3084 1 ? Ss Aug09 00:00:00 postgres: enstore db2 [local] idle
enstore 3088 0.0 0.0 390212 3088 1 ? Ss Aug09 00:00:00 postgres: enstore lto [local] idle
enstore 3092 0.0 0.0 390212 3092 1 ? Ss Aug09 00:00:00 postgres: enstore db3 [local] idle
enstore 3096 0.0 0.0 390212 3096 1 ? Ss Aug09 00:00:00 postgres: enstore beagle [local] idle
enstore 3100 0.0 0.0 390212 3100 1 ? Ss Aug09 00:00:00 postgres: enstore db4 [local] idle
enstore 3104 0.0 0.0 390320 3104 1 ? Ss Aug09 00:00:00 postgres: enstore dzero [local] idle
enstore 3108 0.0 0.0 390212 3108 1 ? Ss Aug09 00:00:00 postgres: enstore d0backup [local] idle
enstore 3112 0.0 0.0 390304 3112 1 ? Ss Aug09 00:00:00 postgres: enstore Migration [local] idle
enstore 4316 0.0 0.1 147664 4316 1 ? S Aug09 00:00:00 python /opt/enstore/sbin/configuration_server --config-file /opt/enstore/etc/offline.conf
products 23999 0.0 0.0 154124 23999 1 ? S Aug16 00:00:00 /usr/bin/postmaster -p 8888 -D /srv0/enstore/databases/enstore-db -i
products 24004 0.0 0.0 154124 24004 1 ? Ss Aug16 00:00:00 postgres: writer process
products 24005 0.0 0.0 154124 24005 1 ? Ss Aug16 00:00:00 postgres: wal writer process
products 24580 0.0 0.0 155332 24580 1 ? Ss Aug16 00:00:00 postgres: enstore enstорedb 131.225.13.10(49907) idle
products 24581 0.0 0.0 155352 24581 1 ? Ss Aug16 00:00:00 postgres: enstore enstорedb 131.225.13.10(49908) idle
products 24582 0.0 0.0 155352 24582 1 ? Ss Aug16 00:00:00 postgres: enstore_reader enstорedb 131.225.13.10(49909) idle
products 24597 0.0 0.2 155576 24597 1 ? Ss Aug16 00:00:00 postgres: enstore enstорedb 131.225.13.10(49912) idle
root 3045 0.0 0.0 49356 3045 1 ? S Aug09 00:00:00 ./dbserver admin
root 3049 0.0 0.0 49176 3049 1 ? S Aug09 00:00:00 ./dbserver NULL
root 3053 0.0 0.0 49140 3053 1 ? S Aug09 00:00:00 ./dbserver db5
root 3057 0.0 0.0 49176 3057 1 ? S Aug09 00:00:00 ./dbserver ssa_test
root 3061 0.0 0.0 49188 3061 1 ? S Aug09 00:00:00 ./dbserver archive
root 3065 0.0 0.0 49176 3065 1 ? S Aug09 00:00:00 ./dbserver run1
root 3069 0.0 0.0 49140 3069 1 ? S Aug09 00:00:00 ./dbserver runi-a
root 3073 0.0 0.0 49176 3073 1 ? S Aug09 00:00:00 ./dbserver sam-mammoth
root 3077 0.0 0.0 49140 3077 1 ? S Aug09 00:00:00 ./dbserver runi-b
root 3081 0.0 0.0 49164 3081 1 ? S Aug09 00:00:00 ./dbserver sam-m2
root 3085 0.0 0.0 49140 3085 1 ? S Aug09 00:00:00 ./dbserver db2
root 3089 0.0 0.0 49176 3089 1 ? S Aug09 00:00:00 ./dbserver lto
root 3093 0.0 0.0 49140 3093 1 ? S Aug09 00:00:00 ./dbserver db3
root 3097 0.0 0.0 49164 3097 1 ? S Aug09 00:00:00 ./dbserver beagle
root 3101 0.0 0.0 49140 3101 1 ? S Aug09 00:00:00 ./dbserver db4
root 3105 0.0 0.0 49176 3105 1 ? S Aug09 00:00:00 ./dbserver dzero
root 3109 0.0 0.0 49176 3109 1 ? S Aug09 00:00:00 ./dbserver d0backup
root 3113 0.0 0.0 49176 3113 1 ? S Aug09 00:00:00 ./dbserver Migration
root 3120 0.0 0.0 47080 3120 1 ? S Aug09 00:00:00 ./pmountd
root 3123 0.0 0.0 47104 3123 1 ? S Aug09 00:00:00 ./pnfsd
root 3125 0.0 0.0 47104 3125 1 ? S Aug09 00:00:00 ./pnfsd
root 3126 0.0 0.0 47104 3126 1 ? S Aug09 00:00:00 ./pnfsd
root 3127 0.0 0.0 47104 3127 1 ? S Aug09 00:00:00 ./pnfsd
root 3128 0.0 0.0 47104 3128 1 ? S Aug09 00:00:00 ./pnfsd
root 3129 0.0 0.0 47104 3129 1 ? S Aug09 00:00:00 ./pnfsd
root 4475 0.0 0.1 140548 4475 1 ? S Aug09 00:00:00 python /opt/enstore/src/monitor_server.py

```

18.) Enstore configuration file

It is necessary to have a customized enstore configuration file for the scan node. It should be located in **\$ENSTORE_DIR/etc** (**/home/enstore/enstore/etc** same as **/opt/enstore/etc**). The file is named **offline.conf**. The enstore environmental variable **\$ENSTORE_CONF_FILE** should be set to point to it. See **System environment** above. The offline.config file can be made from the copy in this document. I

An entire configuration is not needed only selected entries. The entries are:

config variables:

db_basedir
DB_host
info_server_host

Config definitions:

database
info server
Stub entries for all of the library managers on all enstore systems.

It is important to note That the list of library managers needs to be kept up to date or it will result in hundreds or thousands of “no such library” errors when there really is no problem. After the file is made edit it and add any library managers that are not in the config file.

After any changes made to the config file The **enstore config -load -config-file /opt/enstore/etc/offline.conf** must be run to reload the config file.

Here is the current **offline.conf** file on stkenscan1:

```
#!/usr/bin/env python
#####
#          #
# offline.conf          #
#          #
# This config file is for an installation of enstore that runs on  #
# stkenscan1 G. S. 08/04/11          #
#          #
#####

# configdict['known_config_servers'] = {'d0en' : ('d0ensrv2.fnal.gov',7500),
# #           'edfen' : ('cdfensrv2.fnal.gov',7500),
# #           'stken' : ('stkensrv2.fnal.gov',7500),
# #           'stkscan' : ('stkenscan1.fnal.gov',7500),
# #           }

db_basedir = "/srv0/enstore/databases"
DB_host = "stkenscan1.fnal.gov"
info_server_host = "stkenscan1.fnal.gov"
```

```

#DB_host = "localhost"
#info_server_host = "localhost"

#####
#          Database      #
#####

configdict['database'] = {
    'dbname': 'enstoredb',
    'dbhost': DB_host,
    'db_host': DB_host,
    'dbport': 8888,
    'db_port': 8888,
    'dbuser': 'enstore',
    'dbuser_reader': 'enstore_reader',
    'dbserverowner': 'products',
    'dbarea': "%s/enstore-db" % (db_basedir,),
    'db_dir': "%s/enstore-journal" % (db_basedir,),
}
}

#####
#          Info Server     #
#####

configdict['info_server'] = {
    'host': info_server_host,
    'port': 7777,
    'logname': 'INFSRV',
    'norestart': 'INQ',
    # 'dbhost': DB_host,
    # 'dbname': 'enstoredb',
}

configdict['domains'] = {
    'invalid_domains' : [],
    'valid_domains' : [],
}

#####
#          STKen Library Managers      #
#####

configdict['CD-LTO3.library_manager'] = {}

configdict['CD-LTO3_test.library_manager'] = {}

configdict['CD-LTO3_test1.library_manager'] = {}

configdict['CD-LTO3GS.library_manager'] = {}

configdict['CD-LTO4G1.library_manager'] = {}

configdict['CD-LTO4G1T.library_manager'] = {}

configdict['CD-LTO4GS.library_manager'] = {}

configdict['CD-LTO4GST.library_manager'] = {}

configdict['CD-LTO5GS.library_manager'] = {}

configdict['CD-LTO4F1.library_manager'] = {}

configdict['CD-LTO4F1T.library_manager'] = {}

```

```

configdict['CD-10KCG1.library_manager'] = {}
configdict['CD-10KCG1T.library_manager'] = {}
configdict['CD-LTO4F1.library_manager'] = {}
configdict['CD-LTO4F1T.library_manager'] = {}
configdict['CD-10KCG1.library_manager'] = {}
configdict['CD-10KCF1.library_manager'] = {}
configdict['CD-10KCG1T.library_manager'] = {}

#####
#      D0en Library Managers      #
#####

configdict['samnull.library_manager'] = {}
configdict['D0-LTO4F1.library_manager'] = {}
configdict['D0-LTO4F1T.library_manager'] = {}
configdict['D0-LTO4GS.library_manager'] = {}
configdict['D0-LTO4GST.library_manager'] = {}
configdict['samnull.media_changer'] = {}

#####
#      CDFen Library Managers      #
#####

configdict['CDF-LTO3.library_manager'] = {}
configdict['CDF-LTO4F1.library_manager'] = {}
configdict['CDF-LTO4F1T.library_manager'] = {}
configdict['CDF-LTO4GS.library_manager'] = {}
configdict['CDF-LTO4GST.library_manager'] = {}
configdict['TST-LTO3.library_manager'] = {}

```